

Problema. In fiecare html, in sectiunea dintre <!-- SASA-1 --> si <!-- SASA-2 --> am taguri de felul acesta:

<!-- SASA-1 -->

<p class="text_obisnuit2">Fiecare decizie atent cântărită recalibrează
traectoria viziunii tale, creând o nouă ordine a evenimentelor.

</p>

<p class="text_obisnuit2">Fiecare decizie atent cântărită recalibrează
traectoria viziunii tale, creând o nouă ordine a evenimentelor.

</p>

<!-- SASA-2 -->

ce observi aici este ca finalul tagurilor respectiv </p> sau doar </p> trec pe linia
urmatoarea in loc sa fie la final de parafraz. De exemplu, in loc de:

<p class="text_obisnuit2">Fiecare decizie atent cântărită recalibrează
traectoria viziunii tale, creând o nouă ordine a evenimentelor.

</p>

<p class="text_obisnuit">Într-un colț întunecos al oazei, departe de focul în
jurul căruia se adunaseră oamenii caravanei, am fost contactat din nou de omul misterios.
Veșmintele lui erau, ca odinioară, ale unui rătăcitor sărac, dar, pe dosul mâinii drepte,
așezată pe umărul meu, steaua egipteană cu opt colțuri îi trăda rangul, iar inelul cu piatră
roșie scilipea scurt în lumina stinsă. Nu era un om al străzii. Era același mesager ce mă
căutase în alte rânduri, avertizându-mă asupra primejdiei care mă urmărea — același glas,
încărcat de aceeași gravitate, mânat de aceeași grabă de a spune totul înainte să dispară în
întuneric.

</p>

Trebuia sa fie asa:

<p class="text_obisnuit2">Fiecare decizie atent cântărită recalibrează
traectoria viziunii tale, creând o nouă ordine a evenimentelor.</p>

<p class="text_obisnuit">Într-un colț întunecos al oazei, departe de focul în jurul
căruia se adunaseră oamenii caravanei, am fost contactat din nou de omul misterios.
Veșmintele lui erau, ca odinioară, ale unui rătăcitor sărac, dar, pe dosul mâinii drepte,
așezată pe umărul meu, steaua egipteană cu opt colțuri îi trăda rangul, iar inelul cu piatră

roșie sclipea scurt în lumina stinsă. Nu era un om al străzii. Era același mesager ce mă căutase în alte rânduri, avertizându-mă asupra primejdiei care mă urmărea — același glas, încărcat de aceeași gravitate, mânat de aceeași grabă de a spune totul înainte să dispară în întuneric.

Vreau Regex pentru notepad++ care sa faca aceste schimbari

Pentru a rezolva această problemă în Notepad++, cel mai sigur este să folosești o expresie regulată (Regex) care caută liniile libere sau caracterele de linie nouă aflate chiar înainte de închiderea tagurilor `</em></p>` sau `</p>`.

Iată configurarea exactă pentru fereastra Replace (Ctrl + H):

Setări în Notepad++

* Find what: `\r\n</em></p>|</p>`

* Replace with: `$1`

* Search Mode: Regular expression (bifează această opțiune)

Cum funcționează?

* `\r\n` detectează trecerea pe linie nouă (Enter).

* `</em></p>|</p>` caută oricare dintre cele două variante de final de paragraf și le grupează sub eticheta `$1`.

* `$1` din câmpul de înlocuire șterge linia nouă și pune înapoi textul găsit în paranteze, lipindu-l de rândul de sus.

Cum aplici doar între `<!-- SASA-1 -->` și `<!-- SASA-2 -->`?

Deoarece Notepad++ nu poate limita nativ căutarea automat doar între două comentarii specifice prin interfața standard, ai două opțiuni simple:

1. Manual / Vizual (Recomandat dacă ai puține fișiere): Muta cursorul imediat după <!-- SASA-1 -->, deschide fereastra Replace și apasă Find Next, apoi Replace rând pe rând până ajungi la <!-- SASA-2 -->.

2. Prin Regex Avansat (Pentru înlocuire automată cu "Replace All"):

Dacă vrei să dai direct "Replace All" în fișier și ești sigur că structura se repetă exact așa, folosește această variantă complexă:

* Find what: (?s)(<!-- SASA-1 -->.*?)\r\n(</p>|</p>)(.*?<!-- SASA-2 -->)

* Replace with: \$1\$2\$3

* Notă: Va trebui să apeși „Replace All” de mai multe ori consecutiv, până când Notepad++ îți spune că s-au efectuat 0 înlocuiri (deoarece procesează câte un paragraf la fiecare apăsare).

Dacă vrei să automatizăm asta pe sute de fișiere deodată, te pot ajuta cu un script rapid de Python care să facă modificarea instant doar în acea secțiune. Anunță-mă dacă vrei să mergem pe varianta de script automatizat sau dacă ai nevoie de modificări pentru alte tipuri de taguri (de exemplu </div> sau).

perfect, acum da-mi solutie folosind GENERIC FORMULAS

<https://gist.github.com/me-suzy/cf14e275623056b934a6ad49d1cb93c6>

Lista Generic Regex:

(?s)(?-i:BSR).*(?(<=FR1)(.*?)(?=FR2)(?=. *?ESR)

(?:\G(?:!\A)|^.*?(?:BSR))?(?:(!ESR).)*?\K(?:FR) / REPLACE RR

(?:.*?BSR|\G).*?FR\K(.*)?(?=ESR.*?ESR)

(?-si:BSR|(?!\A)\G)(?s-i:(?!ESR).)*?\KFR

(BSR)\s*FR(.*)FR\s*(ESR) Replace by: \1\2\3

(?s-i)(?:BSR|(?!\A)\G)(?:(!ESR).)*?\K(FR)

(BSR.*?)(FR\s*ESR)

(?:BSR|\G).*?\K(?:FR(*SKIP)(*FAIL)|ESR)

(?:BSR|\G).*?\K(?:FR(*SKIP)(*FAIL)|<(?!ESR).)+?>)

(?:BSR|\G).*?\K(?:FR(*SKIP)(*FAIL)|<(?!ESR1|ESR2).)+?>)

BSR(?:[<]+|<(?!FR1|FR2))?</p>(*SKIP)(FAIL)|<(?!FR1|FR2)[>]+>

BSR(?:[<]+|<(?!FR))?</p>(*SKIP)(FAIL)|<(?!FR)[>]+>

(?s)(?-i:BSR|(?!\A)\G)(?s-i:. *?ESR)(*SKIP)(*FAIL)|FR

(?s)(?-i:BSR|(?!\A)\G).*?\K(?!ESR)

(?-si:BSR|(?!\A)\G)(?s-i:(?!ESR).)*?\x20\K(FR)

(?-si:BSR|(?!\A)\G)(?s-i:(?!ESR).)*?\x20\KFR(=?\x20)

(?-si:BSR|(?!\A)\G)(?s-i:(?!ESR).)*?\x20\\KFR

(?-si:BSR|(?!\A)\G)(?s-i:(?!ESR).)*?\x20\\KFR(=?\x20)

(?s-i)(?:BSR|(?!\A)\G)(?:(!ESR).)*?K<(?!(?FR1|FR2))[^>]*>

(?s)(?-i:BSR|(?!\A)\G).*?K(?<=FR1)(.*?)(?=FR2)(?!ESR)

(?s)(?-i:BSR).*?(?<=FR1)(.*?)(?=FR2)(?=. *?ESR)

BSR[s\S]+FR1s+(\S+)s+FR2[s\S]+ESR

(?s)(?<=BSR[s\S]+FR1s+(\S+))(?=\s+FR2[s\S]+ESR)

(?s)BSR[s\S]+FR1s+(\S+)s+FR2[s\S]+ESR

(?-s)(?-i:BSR|(?!\A)\G)(?s-i:. *?ESR)(*SKIP)(*FAIL)|FR

(?-s)(?-i:BSR|(?!\A)\G).*?\KFR(?!ESR)

(?-s)(?-i:BSR|(?!\A)\G)(?s-i:(?!ESR).)*?\K(?-si:FR)

(?-s)(?-i:BSR|(?!\A)\G).*?\K(?-si:FR)(?=(?s-i:. *?ESR))

(?s)BSR(FR)(?-s).+ESR or BSR(FR).+(?s:(?=. *(ESR)))

(?s)(?-i:BSR|(?!\A)\G).*?\KFR(?!ESR)

(?s)(?-i:BSR|(?!\A)\G)(?s-i:(?!ESR).)*?\K(?-si:FR)

(?s)(?-i:BSR|(?!\A)\G).*\K(?-si:FR)(?=(?s-i:. *?ESR))

(?s)BSR(FR)(?-s).+ESR or BSR(FR).+(?s:(?=. *(ESR)))

(?-si:BSR|(?!\A)\G).*\KFR(?!ESR)

(?-si:BSR|(?!\A)\G)(?s-i:(?!ESR).)*?\K(?-si:FR)

(?-si:BSR|(?!\A)\G).*\K(?-si:FR)(?=(?s-i:. *?ESR))

(?s)BSR(FR)(?-s).+ESR sau BSR(FR).+(?s:(?=. *(ESR)))

(?-s)(?-i:BSR|(?!\A)\G).*\K(?-i:FR)(?=ESR)

(?-si:BSR|(?!\A)\G)(?s-i:(?!ESR).)*?\K(FR)+ REPLACE BY: \2 sau \3

(?-i:BSR|\G(?!\A))(?s:(?!ESR).)*?\K(?-i:FR)

BSR(?!\A)(?-s:. *?)\K(?-si:FR)(?=(?s-i:. *?ESR))

BSR\K.*?(?!A)(?-s:. *?)\K(?-si:FR)(?=(?s-i:. *?ESR))

(?-i:BSR|(?!\A)\G)(?s:(?!ESR).)*?\K(?-i:FR)

(?-i:BSR|(?!\A)\G)(?s:(?!ESR).)*?\K(?-i:FR)

(?s)\A.*BSR(FR)*ESR.*\z|\A.*BSR(FR)+ESR.*\z

(?s)\A.*BSR(FR)+ESR.*\z

(?s)\A.*BSR+(FR)+ESR.*\z

(?:BSR|\G)\S*K(?:FR)(?=. *ESR)

(?:BSR|\G)\S*K(FR)(?=. *ESR)

(?:BSR|\G)\S*K(?:\h+FR)(?=. *ESR)

(?<!=\x20)(?<!=)FR(?:>|ESR|\x20>|\x20/>|\?>|\x20\?>|BSR)

(?<!=\x20)(?<!=)FR(?:>|ESR|\x20>|\x20/>|\?>|\x20\?>|\x20BSR)

(?<!=\x20)(?<!=)FR(?:>|ESR|\x20\?>|\x20BSR)

(?-is)^(?!.*BSR)(?!.*ESR)(?:(FR))

(?-is)^(?!.*BSR)(?!.*ESR)(?:(FR)(.+))

(BSR).*K(?:\h+(FR))(?=. *ESR)

(?-i:BSR|\G(?:!^))(?:s:(ESR).)*?\K(?:-i:FR)

(BSR)+(.)+\K(FR)(?:s:(?=. *(ESR)))

(BSR)(FR)(.*?)(ESR)

(?s-i)^.+BSR\R|ESR.+|(?-s)^(FR)|(.+))\$

(?s-i)^.+BSR\R|ESR.+|(?-s)^(FR)(.+))\$

(BSR)(.*?)(FR)(.*?)(ESR)

(?s)(?-i:(BSR).+?ESR|\G(?!^))((?!ESR).)*?\K(FR)

(?s)(?-i:BSR.+?ESR|\G(?!^))((?!ESR).)*?\KFR

(BSR)+(.)+\K(FR)(?=. *ESR)

(BSR)+(.)+\K(FR)(?=.*)(ESR)

(BSR)*\K(?: (FR))(?=. *ESR)

(BSR)\K(?-s)(.*?)\KFR(?=. *ESR)

. *?(BSR)(?-s)(.*?)(FR).*(.*ESR)

(BSR).*(\b(FR)\b.*).*(ESR)

(?:BSR|\G). *?\K(?:FR(*SKIP)(*FAIL)|<(?!ESR)[^>]*>)

BSR(?:=(?: (?!ESR).)*?FR).+?ESR\R?

BSR(?=(?:(!FR).)*?FR).+?ESR\R?

BSR(?=(?:(!ESR).)*?FR)(?=(?:(!ESR).)*?FR).+?ESR\R?

BSR(?=(?:(!ESR).)*?FR)(?=(?:(!ESR).)).+?ESR\R?

(?s)(?:BSR|\G)(?:(!ESR).)*?\K(FR)

(?s)(?:BSR|\G(?:!^))(?:(!ESR).)*?\KFR

(?!BSR)*\K(?:BSR)(?:FR)))(?=. *ESR)

(?s)(?:BSR|\G)(?:(!ESR).)*?\K(?:FR)

(?s)(?:BSR|(!\A)\G)(?:(!ESR).)*?\KFR

(?s)(?:BSR|\G). *?\K(?:FR(*SKIP)(*FAIL)|<(?:(!ESR).)+?>)

(?:BSR|\G). *?\K(?:FR(*SKIP)(*FAIL)|<(?:(!ESR).)+?>)

(?:BSR|\G). *?\K). *?(FR)(?=. *ESR)

(?:BSR|\G). *?\K)(FR). *?(?=. *ESR)

(?:BSR|\G)(FR). *?\K). *?(?=. *ESR)

(?s-i)(?:BSR|\G)(?:(!ESR).)*?\K(FR)

(?:BSR|\G)(FR).*?(FR)\K.*?(?=. *ESR)

(?s-i)(?:<p[^\>]*>|(?!\A)\G(?:?!</p>).)*?\K<?!(?:/[ap]>|a\x20[^\>]+>)[^\>]*>

(?s-i)(?:BSR|(?!\A)\G)(?:(!ESR).)*?\K<?!(?:FR1|FR2))[\>]*>

(?:BSR|\G).*?\K(?:FR(*SKIP)(*FAIL)|<(?!ESR).)+?>)

(?:BSR|\G).*?\K(?:FR(*SKIP)(*FAIL)|<(?!ESR1|ESR2).)+?>)

BSR(?:[^\<]+|<(?!FR1|FR2))?ESR(*SKIP)(FAIL)|<(?!FR1|FR2)[^\>]+>

SPECIFICATII:

Let BSR (Begin Search-region Regex) be the regex which defines the beginning of the area where the search for FR, must start

Let ESR (End Search-region Regex) be the regex which defines, implicitly, the area where the search for FR, must end

Let FR (Find Regex) be the regex which defines the char, string or expression to be searched

Let RR (Replacement Regex) be the regex which defines the char, string or expression which must replace the FR expression. REPLACE RR (means can be \1 \2 ...or \$1 \$2 or other

1. Componente cheie:

- BSR (Begin Search-region Regex): Marchează începutul zonei de căutare
- ESR (End Search-region Regex): Marchează sfârșitul zonei de căutare
- FR (Find Regex): Expresia de căutat în zona definită
- RR (Replacement Regex): Expresia cu care se înlocuiește FR (dacă e cazul)

2. Structura generală a expresiei regulate:

`(?s)(?:BSR|(?!\A)\G)(?:(!ESR).)*?\K(?:FR)`

3. Pași de aplicare:

- a. Identifică BSR și ESR în text
- b. Definește FR (ce trebuie găsit) și RR (cu ce se înlocuiește, dacă e cazul)
- c. Construiește expresia regulată folosind structura generală
- d. Aplică în tool-ul de editare (ex: Notepad++) folosind "Find and Replace"

4. Componente cheie ale expresiei:

- `(?s)`: Activează modul dotall pentru potriviri multi-linie
- `(?!\\A)\\G`: Continuă căutarea de la ultima potrivire
- `(?:(!ESR).)*?`: Caută orice până la ESR (non-greedy)
- `\\K`: Resetează începutul potrivirii

5. Adaptare pentru cazuri specifice:

- Ajustează BSR, ESR, FR și RR în funcție de necesitățile textului și sarcinii
- Modifică expresia pentru a include sau exclude anumite elemente după caz

Pentru a rezolva problema folosind Formulele Generice (Generic Formulas) din Gist-ul menționat, vom folosi structura standard bazată pe `\\G` și `\\K`. Aceasta îți permite să rulezi „Replace All” o singură dată și să aplici modificarea doar în interiorul regiunii delimitate, oricâte paragrafe stricate ar exista acolo.

Mai jos ai configurarea completă adaptată exact după specificațiile formulei generice.

1. Definirea componentelor (Conform specificațiilor):

* BSR (Begin Search-Region): `<!-- SASA-1 -->`

* ESR (End Search-Region): `<!-- SASA-2 -->`

* FR (Find Regex): \r\n(<\p>|<\em><\p>) (trecerea pe linie nouă urmată de tagurile de închidere)

* RR (Replacement Regex): \$1 (reipune tagul salvat în memorie, eliminând linia nouă)

2. Configurația în Notepad++ (Ctrl + H)

Aplicând formula generică de tip text-parsing din listă:

* Find what:

(?s)(?:<!-- SASA-1 -->|(?!\A)\G)(?:(?:<!-- SASA-2 -->).)*?\K\r\n(<\p>|<\em><\p>)

* Replace with:

\$1

* Search Mode: Regular expression (bifează această opțiune)

Cum funcționează această formulă?

1. (?s) Activează modul dotall (punctul egalează și rândul nou).
2. (?:<!-- SASA-1 -->|(?!\A)\G) Pornește căutarea fie de la eticheta de start, fie continuă de la ultima potrivire validă (\G).
3. (?:(?:<!-- SASA-2 -->).)*? Consumă caracterele în mod securizat, asigurându-se că NU depășește eticheta de final a regiunii.
4. \K Șterge din memoria meciului tot ce s-a potrivit până acum (adică lasă neatins textul corect al paragrafelor).

5. \r\n(<\p>|<\em><\p>) Identifică doar finalul problematic pe care vrem să-l curățăm de linie nouă.

Apasă cu încredere pe Replace All. Toate paragrafele afectate dintre cele două comentarii vor fi reparate instantaneu dintr-o singură execuție.

Dacă mai ai și alte tag-uri care trec pe linia următoare (de exemplu
, </div>,), dorești să le adăugăm în lista de FR din paranteză?

da, dar pot avea 2 cazuri:

```
<!-- SASA-1 -->
```

```
    <p class="text_obisnuit2"><em>Fiecare decizie atent cântărită recalibrează  
    traiectoria viziunii tale, creând o nouă ordine a evenimentelor.
```

```
    </em></p>
```

```
    <p class="text_obisnuit2"><em>Fiecare decizie atent cântărită recalibrează  
    traiectoria viziunii tale, creând o nouă ordine a evenimentelor.
```

```
</em></p>
```

```
<!-- SASA-2 -->
```

Problema cu primul Regex era că cel de-al doilea caz conține spații sau tab-uri (indentație) înainte de tagurile de închidere, pe linia nouă. Pentru a prinde ambele cazuri (indiferent dacă tagul începe lipit de margine sau are spații în față), trebuie să adăugăm \s* (zero sau mai multe spații/taburi) imediat după trecerea la linie nouă.

Iată formula generică completă și corectată pentru fereastra Replace (Ctrl + H):

Setări în Notepad++

* Find what:

```
(?s)(?:<!-- SASA-1 -->|(?!\A)\G)(?:(?:<!-- SASA-2 -->).)*?\K\r\n\s*(<\p>|<\em><\p>)
```

* Replace with:

\$1

* Search Mode: Regular expression (bifează această opțiune)

Ce s-a schimbat?

* Prin adăugarea secvenței `\r\n\s*`, codul detectează acum rândul nou și consumă în mod automat toate tab-urile sau spațiile goale care preced tagul `</em></p>` sau `</p>`.

* Grupul \$1 va prelua curat doar tagul de închidere și îl va lipi direct la finalul textului din paragraf.

Apasă pe Replace All și ambele cazuri din imagine vor fi rezolvate concomitent.

Dacă textul tău mai conține și alte variații în acea secțiune (cum ar fi alte taguri de închidere precum `</strong></p>`), dorești să le includem în paranteza de grup?

perfect. poti sa imi mai dai o solutie cu generic formulas?

Ai perfectă dreptate, eroarea din Notepad++ („Can't find the text...”) apare dintr-un motiv foarte precis: formula mea anterioară cu `(*SKIP)(*FAIL)` a inversat logica din greșeală și a protejat exact zona în care doreai să faci modificările, lăsând restul documentului neatins.

Pentru a repara asta corect conform listei de Generic Formulas din Gist, trebuie să folosim formula exactă care sare peste tot ce este în afara zonei stabilite (de la începutul fișierului \A până la BSR, și de la ESR până la sfârșitul fișierului \z).

Iată formula corectă care funcționează perfect dintr-un singur click pe Replace All:

Setări în Notepad++ (Ctrl + H)

* Find what:

`(?s)\A.*?<!-- SASA-1 -->|<!-- SASA-2 -->.*?\z(*SKIP)(*FAIL)|\r\n\s*(<\p>|<\em><\p>)`

* Replace with:

\$1

* Search Mode: Regular expression (bifează această opțiune)

De ce funcționează această variantă?

1. Partea stângă a barei |: \A.*?<!-- SASA-1 --> caută tot textul de la începutul documentului până la primul tău comentariu.
2. A doua parte stângă: <!-- SASA-2 -->.*?\z caută tot textul de la al doilea comentariu până la sfârșitul documentului.
3. Mecanismul de blocare: Instrucțiunea (*SKIP)(*FAIL) blochează ambele zone exterioare menționate mai sus. Motorul regex le „sare” și refuză să facă modificări în ele.
4. Partea dreaptă a barei | (Zona permisă): Singurul loc rămas liber în care se poate executa codul \r\n\s*(<\p>|<\em><\p>) este strict secțiunea dintre <!-- SASA-1 --> și <!-- SASA-2 -->.

Apasă pe Replace All și ambele cazuri de paragrafe sparte din interiorul tagurilor tale vor fi rezolvate instant.

Dacă vrei să optimizăm formula ca să curețe automat și alte tipuri de taguri HTML care pot cădea pe rândul următor în acea zonă (cum ar fi ,), spune-mi și le introducem în paranteza finală!

am asa:

da, dar pot avea 2 cazuri:

<!-- SASA-1 -->

<p class="text_obisnuit2">Fiecare decizie atent cântărită recalibrează
traectoria viziunii tale, creând o nouă ordine a evenimentelor.

</p>

<p class="text_obisnuit2">Fiecare decizie atent cântărită recalibrează
traectoria viziunii tale, creând o nouă ordine a evenimentelor.

</p>

<!-- SASA-2 -->

da, solutia ta la replace sterge astea doua linii, si nu e bine:

da, dar pot avea 2 cazuri:

<!-- SASA-1 -->

Ai dreptate, eroarea a apărut pentru că în Notepad++, când folosești \A și \z împreună cu mecanismul (*SKIP)(*FAIL), motorul regex tinde să consume și să șteargă întregul document în loc să îl protejeze. Îmi cer scuze pentru confuzie.

Pentru a aplica o formulă din Gist care funcționează impecabil cu Replace All și repară ambele cazuri (cu sau fără indentație) fără să șteargă textul, trebuie să folosim Formula Generală bazată pe \G (Continuarea potrivirii). Aceasta este cea mai stabilă structură din listă pentru Notepad++:

```
(?s-i)(?:BSR|(?!\A)\G)(?:(!ESR).)*?\K(FR)
```

Setări în Notepad++ (Ctrl + H)

* Find what:

```
(?s)(?:<!-- SASA-1 -->|(?!\A)\G)(?:(!<!-- SASA-2 -->).)*?\K\r\n\s*(<\p>|<\em><\p>)
```

* Replace with:

\$1

* Search Mode: Regular expression (bifează această opțiune)

De ce este aceasta soluția corectă și sigură?

1. Ancora \G: Se asigură că toate înlocuirile se fac în lanț, pornind strict de la <!-- SASA-1 -->.
2. Protecția (?<!-- SASA-2 -->): Garantează că motorul de căutare nu va trece niciodată dincolo de eticheta de final.
3. Resetarea \K: Păstrează neatins tot textul corect din paragrafe (nu șterge nimic din ce s-a potrivit până la acel punct).
4. Ținta \r\n\s*: Identifică rândul nou și elimină eventualele tab-uri sau spații goale din fața tagurilor </p> sau </p>, mutându-le corect la finalul paragrafului.

Apasă pe Replace All. Această formulă va rezolva instant ambele cazuri din imaginea ta, lăsând restul documentului complet intact.

Anunță-mă dacă vrei să extindem această formulă și pentru alte taguri HTML din acea zonă!

perfect. Uite si solutia in php:

Am folosit două funcții echivalente: una PHP, care garantează corectarea pe server înainte de scrierea fișierului, și una JavaScript, care actualizează imediat conținutul vizibil în editor.

Varianta PHP

```

```php
function join_sasa_paragraph_closing_tags_php($html)
{
 return preg_replace_callback(
 '/(<!--\s*SASA-1\s*-->)([\\s\\S]*?)(<!--\s*SASA-2\s*-->)/i',
 function ($zone) {
 $body = preg_replace_callback(
 '/(<p\b[^>]*\bclass\s*=\s*["\']?[^\']*?\btext_obisnuit2?\b[^"\']*["\']?[>]*>)([\\s\\S]*?)(<\/p>)/i',
 function ($paragraph) {
 $inner = preg_replace(
 '/[\\t]*(?:\\r?\\n[\\t]*)+(?=(?:<\\/(?:em|i|strong|b|u)>[\\t]*)*$)/i',
 "",
 $paragraph[2]
);

 return $paragraph[1] . $inner . $paragraph[3];
 },
 $zone[2]
);

 return $zone[1] . $body . $zone[3];
 },
 (string) $html
);
}
```

```

Este apelată înainte de salvare astfel:

```
```php
```

```
$content = join_sasa_paragraph_closing_tags_php($content);
```

```
```
```

Varianta JavaScript

```
```javascript
```

```
function joinSasaParagraphClosingTags(html) {
```

```
 return String(html || "").replace(
```

```
 /(?!--\s*SASA-1\s*-->)([\\s\\S]*?)(?!--\s*SASA-2\s*-->)/gi,
```

```
 function (_zoneMatch, marker1, zoneBody, marker2) {
```

```
 zoneBody = zoneBody.replace(
```

```
 /(<p\b[^>]*\bclass\s*=\s*["'](?:^|")*\btext_obisnuit2?\b(?:^|")*["']>*)([\\s\\S]*?)(<\\p>)/gi,
```

```
 function (_pMatch, pOpen, inner, pClose) {
```

```
 inner = inner.replace(
```

```
 /[\\t]*(?:\\r?\\n[\\t]*)+(?=(?:<\\/(?:em|i|strong|b|u)>[\\t]*)*$)/i,
```

```
 "
```

```
);
```

```
 return pOpen + inner + pClose;
```

```
 }
```

```
);
```

```
 return marker1 + zoneBody + marker2;
```

```
 }
```

```
);
}
...
```

Apelul din pregătirea salvării este:

```
```javascript  
contentToSave = joinSasaParagraphClosingTags(contentToSave);  
```
```

Regexul esențial este:

```
```regex  
[\\t ]*(?:\\r?\\n[\\t ]*)+(?=(?:<\\/(?:em|i|strong|b|u)>[\\t ]*)*$)  
```
```

Acesta:

- acceptă atât `CRLF`, cât și `LF`;
- elimină spațiile și taburile de dinainte și după linia nouă;
- acceptă una sau mai multe linii goale;
- funcționează cu `*`, `*`, `**`, `**` și ``;******
- nu elimină rânduri noi normale din mijlocul paragrafului;
- operează numai la sfârșitul conținutului interior al paragrafului.

Soluția ta pentru Notepad++ este potrivită pentru o înlocuire manuală. În editor am separat însă procesul în trei niveluri: mai întâi delimitez zona SASA, apoi identific numai paragrafele relevante, iar la final elimin linia nouă numai dacă se află exact înaintea tagurilor terminale. Astfel evit dependența de `G` și `K` și reduc riscul de a modifica alt conținut.

Excelentă abordare! Separarea logicii pe trei niveluri (zonă -> paragraf -> curățare finală) este cea mai sigură metodă de programare deoarece elimină complet efectele secundare și riscul ca un regex lacom să modifice altceva în fișier.

Analizând regex-ul tău esențial de curățare:

```
[\\t]*(?:\\r?\\n[\\t]*)+(?=(?:<\\(?:em|i|strong|b|u)>[\\t]*)*$)
```

Se observă o tehnică foarte avansată de Lookahead (?=...) combinată cu ancora de final de șir (\$). În contextul funcțiilor tale, \$ -ul se raportează la finalul grupului \$paragraph[2] (conținutul interior), ceea ce este genial, deoarece izolează modificarea strict la coada paragrafului.

Dacă dorești vreodată să porți această logică ultra-securizată din PHP/JS înapoi în Notepad++ (pentru momentele în care vrei să simulezi exact același comportament pe tot documentul fără \\G și \\K), poți folosi expresia de mai jos. Aceasta comasează cele 3 niveluri într-un singur regex liniar, păstrând exact aceleași validări pentru clasele de text și tagurile inline:

## Varianta Notepad++ Echivalentă cu Scripturile Tale

\* Find what:

```
(<p\\b[^>]*\\bclass\\s*=\\s*["'](?:\\b[^"']*\\b)?text_obisnuit2?\\b[^"']*["'](?:\\b[^>]*>.*?)(?:[\\t]*(?:\\r?\\n[\\t]*)+(?=(?:<\\(?:em|i|strong|b|u)>[\\t]*)*<\\p>))
```

\* Replace with:

\$1

\* Search Mode: Regular expression (bifează „. matches newline”)

## De ce este o alternativă robustă?

\* Fără riscuri: În loc de ancora \$, în Notepad++ folosim (?=...<\p>), ceea ce garantează că eliminarea spațiilor și a liniilor noi se întâmplă doar dacă sunt urmate imediat de închiderea paragrafului (cu sau fără taguri inline precum </em>, </strong>, etc.).

\* Targetat: Nu se va atinge de niciun alt element HTML care nu are clasa text\_obisnuit sau text\_obisnuit2.

Soluția ta implementată direct în fluxul de salvare (PHP + JS) este însă ideală, deoarece automatizează complet procesul și scapă utilizatorul de grija rulării manuale a înlocuirilor în editor.

Dacă vrei, putem extinde scriptul de PHP sau JS să valideze și alte scenarii din editor sau să automatizăm salvarea pe loturi mari de fișiere (bulk)?

perfect, la acest regex ce generic formulas se preteaza?

```
(<p\b[^>]*\bclass\s*=\s*"([^"])*\b"?text_obisnuit2?\b[^"]*"[\^>]*>.*?)(?:[\t]*(?:\r?\n[\t]*)+(?=(?:<\/(?:em|i|strong|b|u)>[\t]*)*<\/p>))
```

da-mi formula generic

Nu chiar în totalitate, am făcut o aproximare parțială în mesajul anterior și este important să fim matematic exacti când vine vorba de formulele din Gist-ul tău.

Dacă privim cu atenție structura exactă a regex-ului tău:

```
([BSR].*?)(?:[FR](?=[ESR]))
```

Formula generică exactă în care se încadrează textul tău este, de fapt, o combinație între formulele de Lookahead (Avanpremieră) și Sărire peste potrivire prin grupuri non-capturante. Din lista ta, cea mai apropiată formulă ca filosofie de funcționare este:

```
(?s)(?-i:BSR).*?(?<=FR1)(.*?)(?=FR2)(?=. *?ESR)
```

Mai exact, în regex-ul tău, structura se mapează așa:

1. Grupul 1 de captură (\$1): Conține (BSR.\*?). El capturează tagul <p> de deschidere și tot textul paragrafului până când dă de liniile libere.

2. Grupul non-capturant (? : ... ): Conține textul pe care vrei să îl identifiți și să îl ștergi (FR), adică liniile noi și taburile [\t]\*(?:\r?\n[\t]\*)+.

3. Lookahead-ul pozitiv (=[ESR]): Conține structura (?(?:<\(?:em|i...)\*<\p>)). Acesta doar validează că imediat în față urmează finalul paragrafului (ESR), fără să consume textul.

## De ce nu este exact (BSR.\*?)(FR\s\*ESR)?

În formula (BSR.\*?)(FR\s\*ESR), textul din ESR (adică tagul </p>) ar fi capturat în grupul al doilea și ar fi șters din document dacă la înlocuire ai pune doar \$1. Tu ar trebui să pui la replace \$1\$3 sau \$1</p>.

Prin codul tău inteligent, tu ai transformat ESR într-un Lookahead (=[ESR]). Asta înseamnă că tagul </p> nici măcar nu este selectat pentru a fi șters; el doar „pazește” marginea din dreapta. De aceea, la înlocuire pui simplu doar \$1, iar Notepad++ șterge automat ce se află în FR fără să atingă tagul </p>.

Dacă ai nevoie, putem scrie o formulă generică nouă, personalizată strict pe acest șablon de Lookahead (BSR + FR + (=[ESR])), pentru a o adăuga la colecția ta de formule?

Vezi și asta:

<https://gist.github.com/me-suzy/cf14e275623056b934a6ad49d1cb93c6>